Centrum voor Wiskunde en Informatica

Centre for Mathematics and Computer Science

P.J.F. Lucas

Multiple inheritance and exceptions in frame systems

Computer Science/Department of Software Technology Report CS-R8931 August

1989

Centrum voor Wiskunde en Informatica
Centre for Mathematics and Computer Science

P.J.F. Lucas

Multiple inheritance and exceptions in frame systems

Computer Science/Department of Software Technology Report CS-R8931 August

---

The Centre for Mathematics and Computer Science is a research institute of the Stichting Mathematisch Centrum, which was founded on February 11, 1946, as a nonprofit institution aiming at the promotion of mathematics, computer science, and their applications. It is sponsored by the Dutch Government through the Netherlands Organization for the Advancement of Research (N.W.O.).

Multiple Inheritance and Exceptions in Frame Systems

Peter Lucas
Centre for Mathematics and Computer Science
P.O. Box 4079, 1009 AB Amsterdam, The Netherlands

In this paper, the problem of handling contradictory information arising in frame systems with multiple inheritance is investigated. In the kind of frame systems described, a distinction is made between classes of objects and individual objects, called instances. Classes contain type information and initial attribute values, and are organized in a taxonomic hierarchy. The treatment in the paper focusses on inheritance of attribute values from classes to instances of classes. The method presented here basically amounts to recording the reasoning that takes place in a frame taxonomy by means of so-called inheritance chains, and then applying a notion of 'in-betweenness' to decide which attribute values derivable by means of inheritance should be given preference over others. Furthermore, an algorithm for constructing a special kind of spanning tree for the associated directed graph of a frame taxonomy is described, and is proven to be equivalent to the cancelling of inheritance chains by using 'in-betweenness'. It is shown that in an inconsistent frame taxonomy such a spanning tree does not exist.

1980 Mathematics Subject Classification (1985 revision): 68T30.
1987 CR Categories: I.2.4 [Artificial Intelligence]: Knowledge Representation Formalisms and Methods.

Keywords & Phrases: Multiple inheritance, frame-based systems, automated reasoning, knowledge-based systems, graph search.

1. Introduction

The formalism of frames was introduced by M. Minsky in the early seventies as a means for the specification of semantic control in an image-understanding application [Minsky75]. Since then, the formalism has enjoyed an increasing popularity in knowledge-based systems, due to the natural way in which hierarchically structured information can be represented in the formalism. The frame formalism has been further developed into one of the basic building blocks of more recent expert system building tools, such as KEE, Knowledge Craft, and Nexpert Object [Fikes85, KC88, NO88].

Frames provide a formalism for storing knowledge concerning the properties of individual objects and classes of objects. Part of the properties of an individual or class are specified within a frame as reference information to other, more general frames having properties which also concern the given frame. The other, non-referential properties are specified as constant values of so-called *slots* or *attributes*. By means of the reference information, a collection of frames is organized as a directed (acyclic) graph, in which the vertices represent collections of attributes with associated values and the arcs represent the (directed) referential relationships between frames. Such a collection of frames is called a *taxonomic hierarchy*, *frame taxonomy*, or *taxonomy* for short.

The organization of frames in a taxonomy forms the basis of a method of automated reasoning, called *inheritance*. This method effectively manipulates frames in such a way that attribute values applicable to a given frame are retrieved from other frames in the taxonomy using the reference information specified in each frame. Usually, two basic forms of inheritance are distinguished, based on the structure of the graph representation of the taxonomy. In case the graph representation of the taxonomy is tree-like, inheritance is called *single inheritance*. In the more general case of a graph-like taxonomy, one speaks of *multiple inheritance*. A well-known problem in both forms of inheritance is the handling of attributes occurring more than once in a taxonomy with different values, for then conflicts may arise due to the inheritance of mutually exclusive values. It should be noted that in frame-based systems, specifying different values for an attribute at different locations in a taxonomy is considered a legal and natural way for expressing *exceptions* to general knowledge which in most cases

Report CS-R8931
Centre for Mathematics and Computer Science
P.O. Box 4079, 1009 AB Amsterdam, The Netherlands

but the exceptions holds true in the domain of discourse. Hence, a method for dealing with exceptions is required in a frame-based system, yielding a form of nonmonotonic reasoning. Exceptions can easily be dealt with in case of single inheritance in a tree-like taxonomy, since then inheritance can be taken to proceed from a given vertex in the tree towards the root along branches of the tree, stopping as soon as a value for a particular attribute has been obtained. Then, clearly, no problems with mutually exclusive values arise, for the algorithm always finds at most one attribute value. However, the problem is more difficult to solve in the case of multiple inheritance in a general graph-like taxonomy. In order to handle multiple inheritance in graph-like taxonomies with exceptions, the inheritance algorithm has to incorporate a method for deciding which value of an attribute to prefer over others. The resulting algorithm is called *multiple inheritance with exceptions*. In many present-day frame-based systems a method for multiple inheritance with exceptions has been implemented, but what actions to take in order to handle exceptions adequately varies with the particular implementation. Some systems require the knowledge engineer to explicitly specify the attributes that prevail over others [Bobrow88,NO88]. Other systems give preference to certain attribute values over others by considering the corresponding graph representation of a taxonomy as an ordered one [Bobrow83,KC88]. Both solutions have not proven to be satisfactory.

In this paper, we propose a solution to the problem of multiple inheritance of attribute values in frame systems. Our solution is based on the application of a notion of 'in-betweenness' for the cancellation of certain attribute values. This work has been inspired by, and is partly based on earlier work by David Touretzky in clarifying multiple inheritance in semantic nets [Touretzky86]. Furthermore, an algorithm based on the construction of a spanning tree of a graph is presented which offers an equivalent, but more efficient solution to multiple inheritance with exceptions in graph-like taxonomies.

2. Preliminaries and Basic Notions

2.1. Preliminaries

In the frame formalism which will be used in this paper, two kinds of frames are distinguished:

- *Classes*, which are used for the representation of classes of objects, and generally contain information concerning the values attributes may adopt, i.e. type information, and
- *Instances*, which are employed for the representation of individual objects, and do not contain any type information.

It is assumed that classes have much in common with the record datatype as, for example, in the Pascal programming language. Amongst other things, a class comprises a collection of so-called *attribute-type specifications* of the form $x(a:\tau)$, where x is the name of the class, similar to the name of a record type in Pascal, a is the attribute name, similar to a field name in Pascal, and τ denotes the type of the attribute. It is assumed that any attribute only occurs in a single attribute-type specification. The specification of a class not merely involves giving a collection of attribute-type specifications. In addition, a class y may be specified as a *subclass* of some other class z by the expression $y < z$, meaning that the attribute-type specifications that hold for z hold for y as well. Determining the attribute-type specifications that hold for a class may be done by a mechanism called *type inference*.

Example. Consider the knowledge domain of the human cardiovascular system. To describe that every blood vessel either contains oxygen-rich or oxygen-poor blood, and that an artery is a blood vessel having a diameter which is a real number, the following specification suits the purpose:

blood-vessel(blood : {oxygen-rich, oxygen-poor})
artery < blood-vessel
artery(diameter : **real**)

The first expression represents a class named 'blood-vessel', having a single attribute 'blood', which

may adopt an element from the set {oxygen-rich, oxygen-poor} as its value. The second expression specifies that all information concerning attributes in the class 'blood-vessel' also concerns the class 'artery'. Since the class 'artery' is a subclass of the class 'blood-vessel' we also have that artery(blood : {oxygen-rich, oxygen-poor}). On the other hand, we do not have blood-vessel(diameter : **real**). □

An instance x of a particular class y, specified by $x \ll y$, is an object of type y, i.e. $x : y$, for which a collection of attributes with constant values have been filled in. The fact that an attribute a of an instance x has value c will be denoted by $x[a = c]$.

EXAMPLE. Consider again the two classes 'blood-vessel' and 'artery' introduced in the preceding example. The brachial artery is an individual artery, i.e. brachial-artery ≪ artery, containing oxygen-rich blood. When filling in the attributes specified in the preceding example we obtain in this case:

brachial-artery[blood = oxygen-rich]
brachial-artery[diameter = 3]

□

There are many situations in representing a problem domain where it is convenient to fix some of the values of attributes of instances of a class within the class or its superclasses in advance. Such is in particular convenient when the values turn out to be the same for most of the instances of a class. The following example illustrates this idea.

EXAMPLE. Consider the preceding two examples in this section once more. In this case, it appears to be convenient to add the specification

artery[blood = oxygen-rich]

since almost all individual arteries do indeed contain oxygen-rich blood. Repeating this information for each instance of the class 'artery' would lead to introducing redundancy. As a consequence, the specification brachial-artery[blood = oxygen-rich] is no longer required. □

In the sequel, knowledge is assumed to be specified in a kind of frame language as sketched in the foregoing. In this article, we address the problem of dealing with different values of the same attribute specified in different classes, which propagate to the instances of the classes by inheritance. Properties of the subclass relation with respect to the well-typedness of a frame taxonomy, will not be dealt with in the present paper. To keep the treatment of attribute-value inheritance as simple as possible, we shall even disregard the fact that classes contain type information in the remainder of the paper. Only the subclass relation will be preserved as a means of attribute-value inheritance. The interested reader should read [Cardelli84] and [Smolka87] for a formal treatment of multiple inheritance and datatypes.

2.2. Basic notions

In this section some basic notions used in the remainder of the paper are presented. We assume that the reader is familiar with relations on finite sets, with closure properties [Lewis81], and with basic graph theory [Wilson85].

In the sequel, $K = \{y_1, y_2, \ldots, y_n\}$ will denote a fixed set of classes, and $I = \{x_1, x_2, \ldots, x_m\}$ will denote a fixed set of instances; the sets I and K are disjoint, $m \geq 0$, $n \geq 1$. The set of frames F is taken to be $I \cup K$. We now define a language for the specification of frame information, and give some examples of its use.

DEFINITION 2.1. Let K denote the fixed set of classes. The *subclass relation* $< \subseteq K \times K$ is an irreflexive and transitive binary relation on K. For a pair $(x,y) \in <$, denoted by $x < y$, it is said that x *is a subclass of* y. □

It is assumed that K contains a *most general class* denoted by $\top$, where for each $y \neq \top \in K$ we have $y < \top$. Furthermore, complementary to the subclass relation $<$ we have the *superclass relation* $>$; for a pair $(x,y) \in >$, denoted by $x > y$, it is said that x *is a superclass of* y.

DEFINITION 2.2. Let I denote the fixed set of instances and K the fixed set of classes. Then, the *instance-of function* $\ll: I \rightarrow K$ is a mapping from I to K. In the sequel, $\ll(x) = y$ will be denoted by $x \ll y$; it is said that x *is an instance of* y. □

The subclass relation and the instance-of function introduced in the two preceding definitions only describe reference information. The following definition introduces a new relation meant to obtain a complete language for the specification of frame information.

DEFINITION 2.3. Let F be the set of frames such that $F = I \cup K$, where I is the set of instances in F, and K the set of classes in F. Let $A = \{a_1, a_2, \ldots, a_p\}$ be a fixed set of attributes, and let $C = \{c_1, c_2, \ldots, c_q\}$ be a fixed set of constants. Then, a triple $(x,a,c) \in F \times A \times C$, denoted by $x[a = c]$, is called an *attribute-value specification*. The *attribute-value relation* Θ is a ternary relation on F, A and C, i.e. $\Theta \subseteq F \times A \times C$, such that for each $y_i[a_k = c_r]$, $y_j[a_l = c_s] \in \Theta$ we have: if $i = j$ and $k = l$ then $r = s$. □

As stated in the preceding subsection, an attribute-value specification $x[a = c]$ is meant to express that the attribute a has the constant value c in frame x.

EXAMPLE. Consider the set of frames F = {brachial-artery, artery, vein, blood-vessel}, the set of attributes A = {diameter, wall, contains}, and the set of constants C = {3, muscular, blood, fibrous}. Then, all expressions shown below are examples of attribute-value specifications:

brachial-artery [diameter = 3]
artery [wall = muscular]
blood-vessel [contains = blood]
vein [wall = fibrous]
vein [wall = muscular]

Note that, when taken together in an attribute-value relation Θ, only one of the last two attribute-value specifications would be allowed (we would choose the first one of the two). □

DEFINITION 2.4. Let F be the set of frames such that $F = I \cup K$, where I is the set of instances in F, and K the set of classes in F. Let A be the set of attributes, and C the set of constants. Let Θ be the attribute-value relation defined according to Definition 2.3. Then, the *instance-restricted attribute-value relation* $\Theta|_I$, is the set of all $x[a = c] \in \Theta$, such that $x \in I$. Similarly, the *class-restricted attribute-value relation* $\Theta|_K$, is the set of all $y[a = c] \in \Theta$, such that $y \in K$. □

DEFINITION 2.5. Let I be the set of instances, K the set of classes, A the set of attributes, and C the set of constants, where I, K, A and C are disjoint. Let N be the quadruple $N = (I,K,A,C)$. Furthermore, let the relations $<$ and Θ be defined according to the Definitions 2.1, and 2.3, respectively, and the function $\ll$ according to Definition 2.2. Then, a *taxonomy* T is a quadruple $T = (N,\Theta,\ll,<)$. □

EXAMPLE. Let I = {brachial-artery} be the set of instances, and K = {artery, vein, blood-vessel} the set of classes. Furthermore, let A and C be as in the preceding example. Let Θ = {blood-

vessel[contains = blood], artery[blood = oxygen-rich], artery[wall = muscular], brachial-artery[diameter = 3]}, and let the function $\ll$ and the relation $<$ be defined by:

brachial-artery $\ll$ artery
artery $<$ blood-vessel
vein $<$ blood-vessel

Then, $T = (N,\Theta,\ll,<)$ is a taxonomy. Note that neither 'vein $<$ artery' nor 'artery $<$ vein' holds, since these tuples are no part of the relation $<$. □

Informally, a taxonomy $T = (N,\Theta,\ll,<)$ can be depicted as a directed, acyclic graph $G = (V(G),R(G))$, where the vertices in $V(G)$ represent the frames in I and K, and the arcs in $R(G)$ represent either the relation $<$ or the function $\ll$. Each vertex is assumed to have an internal structure representing the collection of attribute-value specifications associated with the frame in the relation Θ. In the graph, an attribute-value specification is depicted next to the vertex it belongs to; only the attribute and constant of an attribute-value specification are shown. The relation $<$ is indicated by a pulled arrow, and the function $\ll$ is depicted by a dashed arrow. In Figure 1 the taxonomy from the previous example is shown.

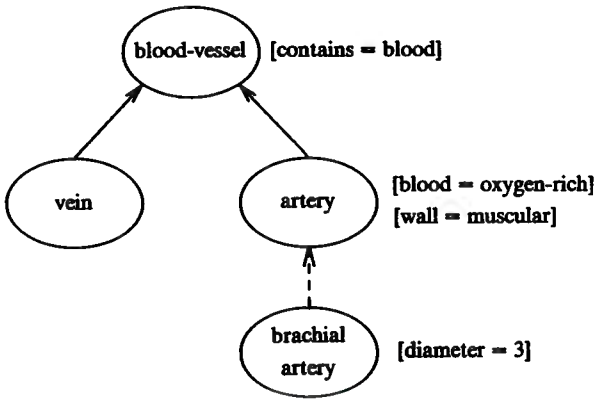

FIGURE 1. Taxonomy consisting of three classes and one instance.

DEFINITION 2.6. Let $T = (N,\Theta,\ll,<)$ be a taxonomy as introduced in the foregoing definition. Then, the *inheritance rule* ρ is the function ρ: $(K \times K) \times (K \times A \times C) \rightarrow (K \times A \times C)$, such that $\rho((x,y),y[a = c]) = x[a = c]$. □

PROPOSITION 2.1. *Let $T = (N,\Theta,\ll,<)$ be a taxonomy, and let $R = \Theta|_K \cup <$. Then, there is a unique minimal set R^ρ, with $R \subseteq R^\rho$, that is closed under the inheritance rule ρ.*

PROOF. Let V be defined as the set of all sets X_i, $i = 1,2,\ldots,$ that are closed under ρ and that have the set R as a subset. Let $R^\rho = \bigcap_{i=1,2,\ldots} X_i$. Now, suppose that $x < y \in R^\rho$ and $y[a = c] \in R^\rho$. Then, for each $X_i \in V$ we have $x < y \in X_i$ and $y[a = c] \in X_i$. Hence, $x[a = c] \in X_i$, since X_i is closed under ρ. It follows that $x[a = c] \in R^\rho$. Hence, R^ρ is closed under ρ. Furthermore, for each $X_i \in V$ we have $R^\rho \subseteq X_i$. We conclude that R^ρ is unique and minimal with respect to set inclusion. □

DEFINITION 2.7. Let $T = (N,\Theta,\ll,<)$ be a taxonomy. Let R^ρ be the unique minimal set as stated in proposition 2.1. Then, the *flattening of the taxonomy T*, denoted by Φ_T, is defined as the set of all attribute-value specifications in R^ρ. □

EXAMPLE. Consider the taxonomy T from the preceding example once more. The flattening of T is simply the class-restricted attribute-value relation $\Theta|_K$, supplemented by the attribute-value specifications computed by ρ. Hence, the flattening of the taxonomy T is equal to the following set Φ_T = {blood-vessel[contains = blood], vein[contains = blood], artery[blood = oxygen-rich], artery[wall = muscular], artery[contains = blood]}. □

It should be noted that another way of computing the flattening of a taxonomy T is obtained by combining the transitivity of $<$ and the inheritance rule ρ.

LEMMA 2.1. Let $T = (N,\Theta,\ll,<)$ be a taxonomy. Let Φ_T be the flattening of T according to Definition 2.7, and let $<$ be the subclass relation. Then, $\Phi_T = \{\rho((x,y),y[a = c]) \mid (x,y) \in <, y[a = c] \in \Theta|_K\} \cup \Theta|_K$.

PROOF. Let us denote the right-hand side of the identity in the lemma by Ψ. On the one hand, let $y[a = c] \in \Psi$ then we have two cases: (a) $y[a = c] \in \Theta|_K$, or (b) $(y,z) \in <$ and $z[a = c]\ \Theta|_K$. In case (a) it follows immediate from Definition 2.7 that $y[a = c] \in \Phi_T$. In case (b) we have pairs $y < v_1$, $v_1 < v_2, \ldots, v_{n-1} < v_n$, $v_n < z$; it therefore follows from the definition of Φ_T that $z[a = c], v_n[a = c], v_{n-1}[a = c], \ldots, v_1[a = c], y[a = c] \in \Phi_T$. Hence, $\Psi \subseteq \Phi_T$.

The other side of the proof is by a symmetrical argument. We conclude that $\Phi_T = \Psi$. □

DEFINITION 2.8. Let $T = (N,\Theta,\ll,<)$ be a taxonomy. Let Φ_T be the flattening of T according to Definition 2.7. Then, the *extension of the taxonomy T*, denoted by E_T, is defined as the set $E_T = \{x[a = c] \mid x \ll y, y[a = c] \in \Phi_T$, provided $x[a = d] \notin \Theta, c \neq d\} \cup \Theta|_I$. □

EXAMPLE. Consider the taxonomy T and the flattening Φ_T in the preceding example. The extension of the taxonomy T is the set E_T = {brachial-artery[contains = blood], brachial-artery[blood = oxygen-rich], brachial-artery[wall = muscular], brachial-artery[diameter = 3]}. □

3. INHERITANCE AND EXCEPTIONS

The relation $<$ defined in the previous section, constitutes the basis for reasoning with frames. In the next subsection we introduce some notions which will be used later in order to formalize multiple inheritance with exceptions. The approach in this subsection is basically to describe inheritance in a taxonomy T by means of so-called inheritance chains. We show that this approach generates a set of attribute-value specifications that under certain conditions equals the flattening of T. Furthermore, in Section 3.2 it is shown that inheritance chains can be employed for dealing with exceptions.

3.1. Inheritance chains and interpretation

One of the notions that plays a central role in this paper is that of inheritance chains. Inheritance chains are used for the representation of the reasoning that takes place in a frame taxonomy. In the following two definitions the form of such chains and a procedure for their construction is described.

DEFINITION 3.1. Let $T = (N,\Theta,\ll,<)$ be a taxonomy, where $N = (I,K,A,C)$. An *inheritance chain* in T is an expression having one of the following forms:

$y[a = c]$
$y_1 < \cdots < y_n$
$y_1 < \cdots < y_{n-1} < y_n[a = c]$

where y, $y_i \in K$, $i = 1, \ldots, n$, $n \geqslant 2$, are classes, and $y[a = c]$, $y_n[a = c] \in \Theta$ are attribute-value specifications. □

Note that attribute-value specifications are allowed only in isolation, or at the end of an inheritance chain.

The set of all possible inheritance chains in a given frame taxonomy can be constructed by the application of the subclass relation.

DEFINITION 3.2. Let $T = (N,\Theta,\ll,<)$ be a taxonomy, where $N = (I,K,A,C)$. Furthermore, let $a \in A$ and $c \in C$. Then, Ω_T is the set of inheritance chains in T such that:

(1) For each $y[a = c] \in \Theta|_K$, we have $y[a = c] \in \Omega_T$.
(2) For each pair $(y_1,y_2) \in <$, we have $y_1 < y_2 \in \Omega_T$.
(3) For each $y_1 < \cdots < y_k \in \Omega_T$ and $y_k < \cdots < y_n \in \Omega_T$, $2 \leqslant k \leqslant n-1$, we have $y_1 < \cdots < y_n \in \Omega_T$, where $y_i \in K$, $i = 1, \ldots, n$, $n \geqslant 3$.
(4) For each $y_1 < \cdots < y_n \in \Omega_T$ and $y_n[a = c] \in \Omega_T$, we have $y_1 < \cdots < y_n[a = c] \in \Omega_T$, where $y_i \in K$, $i = 1, \ldots, n$, $n \geqslant 2$.

□

The transitivity property of the subclass relation will normally not be exploited in our examples, since, as we will see in the sequel, inheritance chains give a characterization at least as powerful. In the next example, we illustrate the construction of inheritance chains.

EXAMPLE. Consider the taxonomy $T = (N,\Theta,\ll,<)$ which is defined such that $I = \{x\}$, $K = \{y_1, y_2, y_3\}$ and $\Theta = \{y_1[a_1 = c_1], y_2[a_2 = c_2], y_3[a_3 = c_3], x[a_4 = c_4]\}$, where a_1, a_2, a_3, and a_4 are distinct attributes, and c_1, c_2, c_3, and c_4 are distinct constants. Assume that $\ll$ and $<$ are defined by: $x \ll y_1$, $y_1 < y_2$, and $y_2 < y_3$. The set Ω_T of inheritance chains in T now consists of the following elements:

$y_1[a_1 = c_1]$
$y_2[a_2 = c_2]$
$y_3[a_3 = c_3]$
$y_1 < y_2$
$y_1 < y_2[a_2 = c_2]$
$y_2 < y_3$
$y_2 < y_3[a_3 = c_3]$
$y_1 < y_2 < y_3$
$y_1 < y_2 < y_3[a_3 = c_3]$

□

A set of inheritance chains is interpreted as a description indicating which attribute-value specifications can *possibly* be inherited by particular classes in the taxonomy. In multiple inheritance with exceptions certain combinations of attribute-value specifications are considered to be 'contradictory'; we will see that under suitable conditions certain inheritance chains can be cancelled, thus preventing the occurrence of such a contradiction. We first introduce the notion of a conclusion of an inheritance chain, which can be viewed as an explicit means for establishing which attribute-value specifications may be inherited from an inheritance chain of a taxonomy.

DEFINITION 3.3. Let $T = (N,\Theta,\ll,<)$ be a taxonomy, where $N = (I,K,A,C)$, and let Ω_T be the set of inheritance chains in T. Let $\Sigma_T = (K \times A \times C) \cup \{\epsilon\}$, where ϵ denotes the *empty attribute-value specification*. Furthermore, let $a \in A$, $c \in C$. The *conclusion function* associated with T is the function γ: $\Omega_T \rightarrow \Sigma_T$, defined as follows:

(1) For each $\omega \equiv x[a = c]$, we have $\gamma(\omega) = \omega$.
(2) For each $\omega \equiv y_1 < \cdots < y_n[a = c]$, $y_i \in K$, $i = 1, \ldots, n$, $n \geqslant 2$, we have $\gamma(\omega) = y_1[a = c]$.
(3) Otherwise, $\gamma(\omega)$ is the empty attribute-value specification, i.e. $\gamma(\omega) = \epsilon$.

We call $\gamma(\omega)$ the *conclusion* of the inheritance chain ω. The image of Ω_T under the application of the function γ is called the *conclusion set* of Ω_T and is denoted by $\gamma(\Omega_T)$. □

In the sequel, we take $\{\epsilon\} \cup X = X$, where X is any set of attribute-value specifications. Note that by this assumption empty attribute-value specifications do not contribute to the conclusion set of a taxonomy.

When the attribute-value specification $z[a = c]$ is obtained as the conclusion of an inheritance chain, we say that the value c of the attribute a is *inherited* by z.

EXAMPLE. Consider again the set Ω_T of inheritance chains from the preceding example. The conclusion set $\gamma(\Omega_T)$ of Ω_T then consists of the following attribute-value specifications:

$$y_1[a_1 = c_1]$$
$$y_2[a_2 = c_2]$$
$$y_3[a_3 = c_3]$$
$$y_1[a_2 = c_2]$$
$$y_1[a_3 = c_3]$$
$$y_2[a_3 = c_3]$$

□

Under certain conditions, the conclusion set $\gamma(\Omega_T)$ of a given set of inheritance chains Ω_T, may contain, amongst other elements, attribute-value specifications which only differ in their specified constant. Let us give an example of such a situation.

EXAMPLE. Consider the following set of inheritance chains Ω_T, containing the following chains:

$$y_1[a_1 = c_1]$$
$$y_1 < y_2[a_1 = c_2]$$

where a_1 is an attribute and c_1, c_2 are distinct constants. The conclusion set $\gamma(\Omega_T)$ contains at least the following attribute-value specifications:

$$y_1[a_1 = c_1]$$
$$y_1[a_1 = c_2]$$

□

Clearly, if a sensible meaning is to be associated with the frame formalism, only one of the conclusions in the foregoing example should hold true. In order to deal with such contradictory information, we introduce the notions of an interpretation and interpreted conclusion set.

DEFINITION 3.4. Let $T = (N,\Theta,\ll,<)$ be a taxonomy, where $N = (I,K,A,C)$. Let Ω_T be the set of inheritance chains in T, and let $\Sigma_T = (K \times A \times C) \cup \{\epsilon\}$, where ϵ is the empty attribute-value

specification. Furthermore, let γ be the conclusion function of T defined according to Definition 3.3. The *interpretation function* of a taxonomy T is the partial function ι_γ: $2^{\Omega_T} \rightarrow 2^{\Sigma_T}$, such that for each $O \in 2^{\Omega_T}$:

$$\iota_\gamma(O) = \begin{cases} \{\gamma(\omega) \mid \omega \in O\} \text{ provided } x\,[a = c_1], x\,[a = c_2], c_1 \neq c_2, \text{ not both in } \gamma(O) \\ \textit{undefined} \qquad \text{otherwise} \end{cases}$$

We call $\iota_\gamma(\Omega_T)$ the *interpreted conclusion set* of T. □

In the sequel, the result of applying the function ι_γ to a set of inheritance chains will be called the result of inheritance.

In the next definition we introduce the notions of consistency and inconsistency of a taxonomy T based on the interpretation function just defined.

DEFINITION 3.5. Let $T = (N,\Theta,\ll,<)$ be a taxonomy, and Ω_T the set of inheritance chains in T according to Definition 3.2. Furthermore, let γ be the conclusion function according to Definition 3.3, and let ι_γ be the interpretation function as defined in Definition 3.4. The taxonomy T is said to be γ-*consistent* if $\iota_\gamma(\Omega_T)$ is defined. Otherwise, T is called γ-*inconsistent*. □

It will be obvious that the taxonomy discussed in the last example is γ-inconsistent. The inconsistency of a taxonomy indicates that the result of inheritance is not unique. We subsequently prove some properties of the mapping ι_γ.

LEMMA 3.1. *Let Ω_U and Ω_V be the sets of inheritance chains of the taxonomies U and V, respectively, such that $\Omega_U \subseteq \Omega_V$. Furthermore, let ι_γ be the interpretation function according to Definition 3.4. If both $\iota_\gamma(\Omega_U)$ and $\iota_\gamma(\Omega_V)$ are defined, then $\iota_\gamma(\Omega_U) \subseteq \iota_\gamma(\Omega_V)$, i.e. the mapping ι_γ is monotonic.*

PROOF. The proof follows straight from the Definitions 3.3 and 3.4. □

In Section 2 we have defined the flattening Φ_T of a taxonomy T. In the following theorem we prove that the flattening Φ_T of a given taxonomy T is equal to the interpreted conclusion set of T, provided that the taxonomy T is γ-consistent.

THEOREM 3.1. *Let $T = (N,\Theta,\ll,<)$ be a taxonomy. Let ι_γ be the interpretation function introduced in Definition 3.4, and let Ω_T be the set of inheritance chains in the taxonomy T. Furthermore, let Φ_T be the flattening of T. Then, if T is γ-consistent, we have $\iota_\gamma(\Omega_T) = \Phi_T$.*

PROOF. Let $R = \Theta|_K \cup <$ and let R^ρ be the unique minimal set with $R \subseteq R^\rho$, that is closed under ρ. From Definition 2.7 we have $\Phi_T \subseteq R^\rho$. Suppose that $y_1 < y_2, \ldots, y_i < y_{i+1}, \ldots, y_{n-1} < y_n \in R^\rho$ and $y_n\,[a = c] \in R^\rho$, then also $y_1\,[a = c] \in R^\rho$, since R^ρ is closed under ρ. From the construction of the inheritance chains in Ω_T in Definition 3.3, it follows that there exists a chain $y_1 < \cdots < y_n\,[a = c] \in \Omega_T$. Since it is given that T is γ-consistent, we have $y_1\,[a = c] \in \iota_\gamma(\Omega_T)$. It follows that each attribute-value specification in R^ρ is a member of $\iota_\gamma(\Omega_T)$. So, $\Phi_T \subseteq \iota_\gamma(\Omega_T)$.

On the other hand, suppose that $y_1 < \cdots < y_n\,[a = c] \in \Omega_T$, then by Definition 3.4, and from T being γ-consistent, we have $y_1\,[a = c] \in \iota_\gamma(\Omega_T)$. From the construction of the inheritance chain $y_1 < \cdots < y_n\,[a = c] \in \Omega_T$, it follows that there exist $y_1 < y_2, \ldots, y_i < y_{i+1}, \ldots, y_{n-1} < y_n$, $y_n\,[a = c] \in R$. It follows that $y_1\,[a = c] \in R^\rho$, since R^ρ is closed under ρ. Hence, it follows that $\iota_\gamma(\Omega_T) \subseteq \Phi_T$. We conclude that $\iota_\gamma(\Omega_T) = \Phi_T$. □

It should be noted that the extension of a γ-consistent taxonomy T as stated in Definition 2.8 can now be redefined in terms of the interpreted conclusion set $\iota_\gamma(\Omega_T)$.

3.2. Exceptions

A taxonomy which is inconsistent in the sense of Definition 3.5 can sometimes be 'made' consistent by cancelling some of the inheritance chains from the entire set of inheritance chains in the taxonomy, by using knowledge concerning the hierarchical ordering of the frames in the taxonomy. As a consequence, certain conclusions from the conclusion set $\gamma(\Omega_T)$ of the taxonomy T are cancelled too, thereby preventing the occurrence of some of the mutually exclusive attribute values which would arise otherwise. However, the interpretation function ι_γ defined in Definition 3.4 does not provide such means for removing conclusions, since this mapping is monotonic. Hence, another interpretation mapping is required. Such a mapping that is capable of dealing with exceptions is developed in this section. This introduces nonmonotonic reasoning within the frame formalism.

In the following definition, the notion of an exception is described more precisely.

DEFINITION 3.6. Let $T = (N,\Theta,\ll,<)$ be a taxonomy, and let Ω_T be the set of inheritance chains in T. Then, every $x[a = c_1] \in \Theta$ for which there exists an inheritance chain $\omega \in \Omega_T$, such that $\gamma(\omega) = x[a = c_2]$, $c_1 \neq c_2 \in C$, is called an *exception*. □

The basic device for handling exceptions is provided by the notion of an intermediary class, which is introduced in the following definition.

DEFINITION 3.7. Let $T = (N,\Theta,\ll,<)$ be a taxonomy. Let Ω_T be the set of inheritance chains in T. A class $y \in K$ is called *intermediary* to an inheritance chain $y_1 < \cdots < y_n \in \Omega_T$, $y_i \in K$, $i = 1, \ldots, n$, $n \geqslant 2$, if one of the following conditions is satisfied:

(1) $y = y_i$ for some i, $1 \leqslant i < n$, or

(2) there exists a chain $y_1 < \cdots < y_p < z_1 < \cdots < z_m < y_q \in \Omega_T$, for some p, q, $1 \leqslant p < q \leqslant n$, such that $y = z_k$, $1 \leqslant k \leqslant m$, where $z_i \neq y_j$, and z_i, $y_j \in K$, $i = 1, \ldots, m$, $m \geqslant 1$, $j = 1, \ldots, n$.

□

EXAMPLE. Consider the taxonomy $T = (N,\Theta,\ll,<)$, where $I = \varnothing$, $K = \{y_1, y_2, y_3, y_4\}$, Θ is empty, and the relation $<$ is defined as follows: $y_1 < y_2$, $y_1 < y_3$, $y_3 < y_4$, $y_2 < y_3$. The set of inheritance chains in T contains amongst other chains the following ones:

$$y_1 < y_3 < y_4$$
$$y_1 < y_2 < y_3$$

The class y_2 is intermediary to both chains. The graphical representation of the taxonomy is given in Figure 2. □

Intermediary classes are applied in the sequel to deal with the occurrence of exceptions in multiple inheritance. By means of intermediary classes some of the inheritance chains are cancelled thus yielding a different set of conclusions of the taxonomy. Such cancellation of inheritance chains is called *preclusion*, and is defined more precisely below.

DEFINITION 3.8. Let $T = (N,\Theta,\ll,<)$ be a taxonomy. Let Ω_T be the set of inheritance chains in T, $a \in A$, and c_1, $c_2 \in C$. A chain $y_1 < \cdots < y_n[a = c_1] \in \Omega_T$, $n \geqslant 1$, is said to *preclude* a chain $y_1 < \cdots < y_m[a = c_2] \in \Omega_T$, $c_1 \neq c_2$, $m > 1$, if y_n is intermediary to $y_1 < \cdots < y_m$. □

EXAMPLE. Consider the set of inheritance chains Ω_T which consists of the following elements:

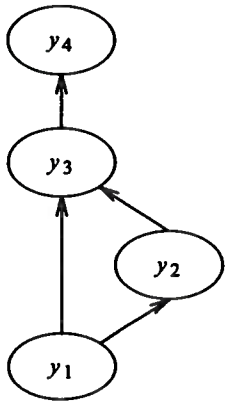

FIGURE 2. A taxonomy with an intermediary class.

ω_1: $y_1 < y_2$
ω_2: $y_1 < y_3$
ω_3: $y_1 < y_2 < y_3$
ω_4: $y_1 < y_2 [a = c_1]$
ω_5: $y_1 < y_3 [a = c_2]$
ω_6: $y_1 < y_2 < y_3 [a = c_2]$

In these chains we have that a is an attribute, and c_1 and c_2 are two distinct constants. The inheritance chain ω_4 precludes both chains ω_5 and ω_6, since y_2 is intermediary to the chains ω_2 and ω_3. □

The notion of preclusion is subsequently used for introducing a new notion of a conclusion of an inheritance chain. A conclusion of an inheritance chain is now defined with respect to a set of inheritance chains.

DEFINITION 3.9. Let $T = (N,\Theta,\ll,<)$ be a taxonomy. Let Ω_T be the set of inheritance chains in T, and γ the conclusion function according to Definition 3.3. Let $\Sigma_T = (K \times A \times C) \cup \{\epsilon\}$, where ϵ denotes again the empty attribute-value specification. The *preclusive conclusion function* is the mapping π: $\Omega_T \times 2^{\Omega_T} \rightarrow \Sigma_T$, such that $\pi(\omega,O) = \epsilon$, if there exists a chain $\omega' \in O$, with $O \subseteq \Omega_T$, such that ω is precluded by ω'; otherwise, $\pi(\omega,O) = \gamma(\omega)$, $\omega \in \Omega_T$. Let $O \subseteq \Omega_T$, then the set $\{\pi(x,O) \mid x \in O\}$ is called the *preclusive conclusion set* of O, and is denoted by Π_O. □

By using the preclusive conclusion function π instead of the conclusion function γ, the notion of an interpretation is redefined, essentially by replacing the latter conclusion function in Definition 3.4 by the former one. The resulting definition is the following.

DEFINITION 3.10. Let $T = (N,\Theta,\ll,<)$ be a taxonomy. Let π be the preclusive conclusion function according to Definition 3.9. Let Ω_T be the set of inheritance chains in T, and let $\Sigma_T = (K \times A \times C) \cup \{\epsilon\}$. The *preclusive interpretation function* of a given taxonomy T is a partial function ι_π: $2^{\Omega_T} \rightarrow 2^{\Sigma_T}$, such that for each $O \in 2^{\Omega_T}$:

$$\iota_\pi(O) = \begin{cases} \{\pi(\omega,O) \mid \omega \in O\} \text{ provided } x[a = c_1], x[a = c_2], c_1 \neq c_2, \text{ not both in } \Pi_O \\ \textit{undefined} \qquad\qquad \text{otherwise} \end{cases}$$

We call $\iota_\pi(\Omega_T)$ the *inheritable conclusion set* of T. □

EXAMPLE. Consider the taxonomy $T = (N,\Theta,\ll,<)$, where the set I is empty, and $K = \{y_1,y_2,y_3,y_4\}$. Let the following attribute-value relation Θ be given:

(1) $y_2[a = c_1]$
(2) $y_3[a = c_2]$

where a is an attribute, and c_1 and c_2 are distinct constants. Furthermore, the relation $<$ is defined by the following elements:

(3) $y_1 < y_2$
(4) $y_1 < y_3$
(5) $y_2 < y_4$
(6) $y_3 < y_4$

This taxonomy is depicted graphically in Figure 3.

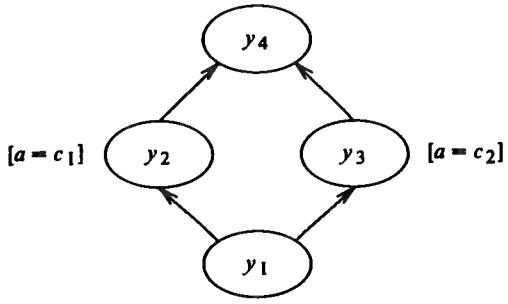

FIGURE 3. A taxonomy having no inheritable conclusion set.

The preclusive conclusion set of the set of inheritance chains in this taxonomy is equal to $\{y_1[a = c_1],\ y_1[a = c_2],\ y_2[a = c_1],\ y_3[a = c_2]\}$. So, the inheritable conclusion set of T is undefined. □

From the foregoing example and the previously defined notion of 'inheritable conclusion set', it follows that the application of multiple inheritance in a taxonomy might still lead to the derivation of contradictory information. Such a taxonomy will be called π-inconsistent according to the following definition.

DEFINITION 3.11. Let $T = (N,\Theta,\ll,<)$ be a taxonomy, and let Ω_T be the set of inheritance chains in T. Furthermore, let π be the preclusive conclusion function according to Definition 3.9, and let $\iota_\pi(\Omega_T)$ be the inheritable conclusion set of T according to Definition 3.10. The taxonomy T is said to be *π-consistent* if $\iota_\pi(\Omega_T)$ is defined; otherwise T is said to be *π-inconsistent*. □

EXAMPLE. Consider the taxonomy $T = (N,\Theta,\ll,<)$, where $I = \{x\}$ is the set of instances, and $K = \{y_1,y_2,y_3\}$ the set of classes. Furthermore, let Θ consist of the following attribute-value specifications:

(1) $x[a_1 = c_1]$
(2) $y_1[a_2 = c_2]$

(3) $y_2[a_3 = c_3]$
(4) $y_3[a_3 = c_4]$

Herein a_1, a_2 and a_3 are distinct attributes and c_1, c_2, c_3 and c_4 are distinct constants. In addition, the function $\ll$ and the relation $<$ are defined as follows:

(5) $y_1 < y_2$
(6) $y_1 < y_3$
(7) $y_2 < y_3$
(8) $x \ll y_1$

This taxonomy is graphically depicted in Figure 4.

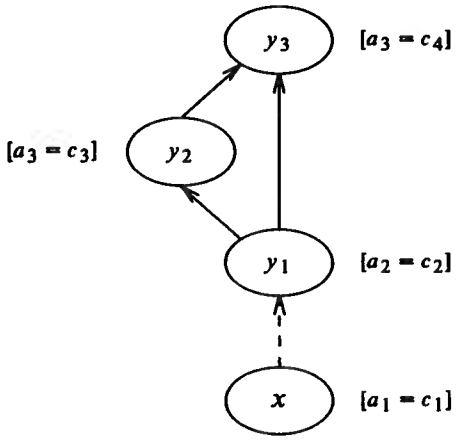

FIGURE 4. A taxonomy having an inheritable conclusion set.

In addition to the inheritance chains 2 to 7 given above, Ω_T contains the following elements:
(9) $y_1 < y_2 < y_3$
(10) $y_1 < y_2[a_3 = c_3]$
(11) $y_1 < y_3[a_3 = c_4]$
(12) $y_2 < y_3[a_3 = c_4]$
(13) $y_1 < y_2 < y_3[a_3 = c_4]$

The conclusion set of this set of inheritance chains Ω_T is equal to $\gamma(\Omega_T) = \{y_1[a_2 = c_2], y_1[a_3 = c_3], y_1[a_3 = c_4], y_2[a_3 = c_3], y_2[a_3 = c_4], y_3[a_3 = c_4]\}$; the interpreted conclusion set $\iota_\gamma(\Omega_T)$ appears to be undefined. Hence, the taxonomy is γ-inconsistent.

Consider the set Ω_T once more, and let us investigate which attribute-values specifications are inheritable under the preclusive conclusion function π. As stated in the definition, an inheritance chain ω is inheritable if not precluded by any other chain. Because only chains ending in an attribute-value specification can be precluded by another chain also ending in an attribute-value specification, the examination of chains of such form will suffice. Hence, let us look at the following chains.

(2) $y_1[a_2 = c_2]$
(3) $y_2[a_3 = c_3]$
(4) $y_3[a_3 = c_4]$
(10) $y_1 < y_2[a_3 = c_3]$
(11) $y_1 < y_3[a_3 = c_4]$
(12) $y_2 < y_3[a_3 = c_4]$
(13) $y_1 < y_2 < y_3[a_3 = c_4]$

Chain 11 is precluded by chain 10, for y_2 is an intermediary class in $y_1 < y_3$; chain 12 is precluded by 3. The reader may verify that the remaining chain $y_1 < y_2 < y_3[a_3 = c_4]$ is precluded too. The inheritable conclusion set $\iota_\pi(\Omega_T)$ of the set of inheritance chains is equal to: $\iota_\pi(\Omega_T) = \{y_1[a_2 = c_2],\ y_1[a_3 = c_3],\ y_2[a_3 = c_3],\ y_3[a_3 = c_4]\}$. Therefore, the taxonomy T is π-consistent. □

From Definition 3.10 is follows that different taxonomies may give rise to identical inheritable conclusion sets. In the sequel, we shall encounter several examples of taxonomies where this is the case. By the following definition, such taxonomies will be called equivalent.

DEFINITION 3.12. Let $T_1 = (N_1,\Theta_1,\ll_1,<_1)$ and $T_2 = (N_2,\Theta_2,\ll_2,<_2)$ be two taxonomies. These taxonomies are called *equivalent* if both T_1 and T_2 are π-inconsistent, or both are π-consistent and we have that $\iota_\pi(\Omega_{T_1}) = \iota_\pi(\Omega_{T_2})$. □

To conclude this section, we introduce the notion of an inheritable extension of a taxonomy T, as an alternative to the definition of an extension introduced in Section 2. Here, only those attribute-value specifications inheritable according to the preclusive interpretation function ι_π, supplemented with the attribute-value specifications in the instances, will occur in the inheritable extension of T.

DEFINITION 3.13. Let $T = (N,\Theta,\ll,<)$ be a taxonomy, and let Ω_T be the set of inheritance chains in T. Furthermore, let π be the preclusive conclusion function according to Definition 3.9, and let $\iota_\pi(\Omega_T)$ be the inheritable conclusion set of T according to Definition 3.10. Then, the *inheritable extension* of T, E_T^ι is defined as the set: $E_T^\iota = \{x[a = c] \mid x \ll y,\ y[a = c] \in \iota_\pi(\Omega_T)$, provided $x[a = d] \notin \Theta$, $c \neq d\} \cup \Theta|_I$. □

EXAMPLE. Consider again the taxonomy T in the preceding example. The inheritable extension of T in this case is equal to $E_T^\iota = \{x[a_1 = c_1],\ x[a_2 = c_2],\ x[a_3 = c_3]\}$. □

4. SPANNING TREES AS A BASIS FOR AN ALGORITHM OF MULTIPLE INHERITANCE

In the preceding section we have been dealing with multiple inheritance with exceptions in frame taxonomies. It was mentioned that these taxonomies could also be represented as acyclic directed graphs. As has been stated in the introduction, single inheritance in tree-like taxonomies, where attributes are allowed to be specified more than once, is easy to implement, contrary to the handling of exceptions in the more general graph-like taxonomies. Informally speaking, the algorithm of single inheritance operates on a tree structure, and traverses a single branch of the tree towards the root, until a value for a particular attribute has been obtained; all other attribute values which might be obtained by going further up towards the root are disregarded.

In this section, we shall study multiple inheritance with exceptions with regard to its algorithmic properties. The structure of this section is as follows. To begin with, in Section 4.1 several notions defined in the two preceding sections will be reformulated into graph-theoretical terms. In Section 4.2 we employ these notions for analysing single inheritance with exceptions in (tree-like) taxonomies. The results in this section will be subsequently utilized in Section 4.3, where we show that it is possible to transform any π-consistent graph-like taxonomy into an equivalent tree-like taxonomy. Moreover, it turns out that single inheritance in the tree representation of the taxonomy yields a set of results that corresponds to the inheritable conclusion set of the original graph-like taxonomy.

4.1. Graph representation of a taxonomy and inheritance chains

We first introduce the graph representation of a taxonomy T, which will be used intensively in the remainder of the paper. For simplicity reasons, we only deal with the graph representation of classes in a taxonomy.

DEFINITION 4.1. Let $T = (N,\Theta,\ll,<)$ be a taxonomy. Furthermore, let A be the set of attributes in T. The *directed attribute graph* $G_a = (V(G_a),R(G_a),\lambda_a)$, or *i-graph* for short, of the taxonomy T is a labelled acyclic directed graph for some attribute $a \in A$, which is obtained from T as follows:

(1) $V(G_a) = \{y \mid y \in K\}$
(2) $R(G_a) = \{(y_i,y_j) \mid y_i < y_j\}$
(3) λ_a: $V(G_a) \rightarrow C$ is a partial labelling function, defined as follows:

$$\lambda_a(y) = \begin{cases} c & \text{if } y[a = c] \in \Theta|_K \\ \textit{undefined} & \text{otherwise} \end{cases}$$

As an abbreviation, $\lambda_a(y) = c$ will also be denoted by $y^{a=c}$, and we say that c is the value of attribute a in y. □

From this definition it follows that a taxonomy T is represented by a set of directed attribute graphs $\{G_a \mid a \in A\}$. Note that Definition 4.1 introduces a one-to-one correspondence between the definition of a taxonomy with respect to a particular attribute $a \in A$, and its directed attribute graph. In particular, under this isomorphism we have that $y^{a=c} \equiv y[a = c]$. Therefore, we shall use part of the terminology introduced in the previous section also in the current context of directed attribute graphs. In the sequel, the notation $y^{a=c}$ will not only be used to denote elements $y \in V(G_a)$ for which we have $\lambda_a(y) = c$, but also to indicate that a particular vertex $y \in V(G_a)$ has inherited an attribute-value pair (a,c). This is similar to the way attribute-value specifications have been defined in Section 2. Note that the directed attribute graphs G_a of a taxonomy T are always connected graphs, which is a consequence of the presence of a most general class $\top$ in K. We call the vertex that corresponds to the most general class the *source* of the graph.

Next, we employ the foregoing definition of a graph representation of a taxonomy in an algorithm that constructs sequences of vertices in a way closely resembling the construction of inheritance chains in Definition 3.2. The following algorithm essentially restates Definition 3.2, but in such a way that its algorithmic properties can be studied more readily.

ALGORITHM 4.1. Let $T = (N,\Theta,\ll,<)$ be a taxonomy, and let $G_a = (V(G_a),R(G_a),\lambda_a)$ be the directed attribute graph representation of T, for some $a \in A$. A sequence of vertices, possibly having an attribute-value specification at the end, will be denoted by $<y_1,\ldots,y_n>$, and $<y_1,\ldots,y_{n-1},y_n^{a=c}>$, respectively. Such sequences will be used for the representation of inheritance chains. To prevent introducing new terminology, the representation of an inheritance chain by means of a sequence is again called an inheritance chain. The function call `Length(s)` gives the number of elements in the sequence s. The function call `Last(s)` yields the last element in the sequence s, e.g. `Last`$(<y_1,\ldots,y_{n-1},y_n^{a=c}>) = y_n^{a=c}$, and the function call `First(s)` returns the first element of the sequence s. Furthermore, `AdjacentFrom(u)` yields all vertices adjacent from u in G_a, i.e. if $(u,v) \in R(G_a)$, then $v \in$ `AdjacentFrom(u)`. The procedure call `Append(s,x)` expresses that the element x is appended to the end of the sequence s. Let s be defined as $<y_1,\ldots,y_{n-1},y_n^{a=c}>$; the procedure call `Conclusion(s)` will then yield the element $y_1^{a=c}$.

We first consider the algorithm for constructing inheritance chains, with as input the attribute graph G_a and as output the set of sequences S.

```
proc ConstructChains(G_a)
global S;

  S ← ∅;
  for each y ∈ V(G_a) do
    if λ_a(y) = c then
      S ← S ∪ {y^{a=c}}
    fi;
    ExtendChain(y,<y>)
  od
end
```

The next procedure yields all possible sequences for a particular vertex, by extending the one-element sequence constructed in the preceding procedure, in a recursive manner.

```
proc ExtendChain(y,s)

  U ← AdjacentFrom(y);
  for each z ∈ U do
    s' ← Append(s,z);
    if λ_a(z) = c then
      s'' ← Append(s,z^{a=c})
    fi;
    S ← S ∪ {s'} ∪ {s''};
    ExtendChain(z,s')
  od
end
```

The following procedure `GammaConsistency` checks whether or not the taxonomy which is represented by the attribute graph G_a is γ-consistent with respect to a given attribute $a \in A$. To this end, the procedure examines all sequences in the set S produced by the procedure `ConstructChains` above. The output of the procedure is the interpreted conclusion set C, or the message that the original taxonomy T is γ-inconsistent.

```
proc GammaConsistency(S)

  C ← ∅;
  for each s ∈ S do
    o ← Conclusion(s);
    if o ≡ y^{a=c} and there exists some y^{a=d} ∈ C, c ≠ d then
      report "taxonomy is γ-inconsistent";
      return
    else
      C ← C ∪ {y^{a=c}}
    fi
  od
end
```

The following procedure `PiConsistency` examines the constructed set of vertex sequences S, and reports whether or not the taxonomy T represented by the attribute graph is π-consistent. The output of this procedure is the inheritable conclusion set C, or the message that the original taxonomy is π-inconsistent.

```
proc PiConsistency(S)
  C ← ∅;
  for each s ∈ S do
    precluded ← false;
    if Last(s) ≡ y^{a=c} and Length(s) ≥ 2 then
      for each s' ∈ S, s' ≠ s while not precluded do
        precluded ← Precludes(s,s')
      od
    fi;
    if not precluded then
      o ← Conclusion(s);
      if o ≡ y^{a=c} and there exists some y^{a=d} ∈ C, c ≠ d then
        report "taxonomy is π-inconsistent";
        return
      else
        C ← C ∪ {y^{a=c}}
      fi
    fi
  od
end
```

The function `Precludes` investigates whether or not `First(s)` = `First(s')`, `Last(s')` $\neq y^{a=d}$ is contained in s, and whether s' contains a vertex y for which we have $\lambda_a(y) = d$, $d \neq c$. If all three conditions are satisfied, the value *true* is returned; otherwise the function returns the value *false*. □

The correctness of these algorithms is easily proven from the definitions in Section 3. We next investigate the worst-case time complexity of the algorithms given above.

LEMMA 4.1. *Let $T = (N,\Theta,\ll,<)$ be a taxonomy. Furthermore, let $G_a = (V(G_a),R(G_a),\lambda_a)$ be the attribute graph representation of the taxonomy. Let $|V(G_a)| = n$, then using Algorithm 4.1 for checking the γ-consistency of a taxonomy takes at most $O(n2^n)$ steps, and checking the π-consistency of a taxonomy requires at most $O(n2^{2n})$ steps.*

PROOF. The procedure `ConstructChains` enumerates all sequences (inheritance chains) for each vertex $y \in V(G_a)$. Consequently, the maximum number of sequences to be generated equals $2 \cdot \sum_{k=1}^{n} \binom{n}{k} - n = 2^{n+1} - n - 2$. The procedure `GammaConsistency` produces an interpreted conclusion set C, or reports γ-inconsistency, just by taking every element $s \in S$, and comparing its conclusion o with each element in the conclusion set C produced so far. Note that the maximal cardinality of C is equal to n, since then each vertex in G_a has an attribute a which has obtained a value. So, once n element have been inserted into C, every subsequently produced attribute-value specification $y^{a=c}$ has to be compared with at most n elements. Checking γ-consistency, and producing the interpreted conclusion set therefore takes at most $\sum_{k=1}^{n-1} k + n(2^{n+1} - 2n - 2) = O(n2^n)$ steps. For checking the π-consistency of a taxonomy, each sequence s having a length greater than or equal to 2, and an attribute-value specification at the end, is compared to all other elements in S to decide whether or not is must be precluded. Checking whether some sequence s' precludes the given sequence s in the procedure `Precludes` takes at most $O(n)$ time. This procedure is executed at most $(2^{n+1} - n - 3)(2^n - n - 1)$ times. Hence, checking for each sequence s whether it can be precluded by a sequence s' takes at most $O(n2^{2n})$ time. □

From the proof of Lemma 4.1 we conclude that checking γ- and π-consistency of a taxonomy is predominated by operations on the representation of inheritance chains. An improvement of the time complexity of the algorithm can only be obtained from abandoning using inheritance chains as a basis for checking consistency. Checking the γ-consistency of a taxonomy can indeed be performed much more efficiently than described in the preceding algorithm, by using another method than offered by inheritance chains. One suitable algorithm starts with the source of the attribute graph G_a. Then, the attribute graph is traversed in a direction reverse to the one indicated by the arcs in $R(G_a)$, where at the same time attribute values are propagated. This way, each arc has to be traversed only once. The taxonomy is established to be γ-inconsistent if and only if at a particular vertex an attribute has obtained more than one attribute value. Let $|V(G_a)| = n$, then the maximal number of arcs in any acyclic directed graph is equal to $n(n-1)/2$. Hence, the worst-case time complexity of the algorithm described is $O(n^2)$.

It is not so easy to improve checking π-consistency in a taxonomy, since considering the relative position of vertices in a graph with respect to other vertices, is an essential aspect of checking π-consistency. The remainder of this paper will be devoted to developing an improved algorithm for checking π-consistency. The starting point for this algorithm is single inheritance in a tree-like taxonomy. This will be dealt with in the next section. In Section 4.3 we then will develop a polynomial time-bounded algorithm for transforming the directed attribute graph of a graph-like taxonomy into an attribute graph of an equivalent tree-like taxonomy. The results in the next section will be part of the overall result achieved in Section 4.3.

4.2. Single inheritance

Single inheritance presupposes that the directed attribute graph on which it operates has the form of a directed tree. This will be the point of departure for defining a taxonomy that has a corresponding attribute graph which is a tree.

DEFINITION 4.2. Let $T = (N,\Theta,\ll,<)$ be a taxonomy. We call the taxonomy T a *tree-like taxonomy* if its associated directed attribute graph G_a, for each $a \in A$, has the following properties:

(1) There is exactly one vertex, called the *root*, with zero out-degree.
(2) Every vertex except the root has an out-degree equal to one.
(3) There is a path from each vertex in G_a to the root.

The directed attribute graph G_a of a tree-like taxonomy is called an *i-tree*. □

ALGORITHM 4.2. Let $T = (N,\Theta,\ll,<)$ be a tree-like taxonomy, and let $G_a = (V(G_a),R(G_a),\lambda_a)$ be the *i*-tree of T, for some $a \in A$. The following (trivial) algorithm for single inheritance with exceptions yields the attribute value of an attribute $a \in A$, for given vertex $y \in V(G_a)$, by searching the *i*-tree G_a.

```
func Inherit(y,w)
  if λa(y) = c
  then
    return(w^(a=c))
  elsif (y,z) ∈ R(Ga)
  then return(Inherit(z,w))
  else return(nil) fi
end
```

The invocation `Inherit(y,y)` only returns a single attribute value for a particular vertex y and the attribute a, or **nil**. □

The value for each $a \in A$, and given vertex y can (naively) be collected by repeatedly executing

Algorithm 4.2 for each $a \in A$. The algorithm gives a feasible procedure for computing the inheritable conclusion set of a tree-like taxonomy T: its worst-case time complexity is quadratic in the number of vertices, for a given attribute $a \in A$. Note that we did not employ inheritance chains in this algorithm, which explains its favourable time complexity, as was also true for checking γ-consistency in the preceding section. Note also that a tree-like taxonomy can never be π-inconsistent.

We subsequently prove that the algorithm above produces results that correspond to the inheritable conclusion set of a tree-like taxonomy, introduced in Definition 3.10.

LEMMA 4.2. Let $T = (N,\Theta,\ll,<)$ *be a tree-like taxonomy, where* $N = (I,K,A,C)$. *Let* Ω_T *be the set of inheritance chains in T, and let A be the set of attributes in T. Then, the set of all elements* $y^{a=c}$, *obtained by applying Algorithm 4.2 on the set of i-trees* $\{G_a \mid a \in A\}$, *for each* $y \in V(G_a)$, *corresponds to* $\iota_\pi(\Omega_T)$.

PROOF. Let $G_a = (V(G_a),R(G_a),\lambda_a)$ be an arbitrary i-tree associated with the taxonomy T according to Definition 4.2. Let $\{c_{i_1},c_{i_2},\ldots,c_{i_m}\} \subseteq C$. We denote the set of attribute-value specifications which results from applying Algorithm 4.2 to each $y \in V(G_a)$ by S_{G_a}. On the one hand, let $y_1^{a=c_{i_k}} \in S_{G_a}$. Then we have either (a) that $\lambda_a(y_1) = c_{i_k}$, or (b) that there exists a path $y_1,y_2,\ldots,y_n$ in G_a, such that $\lambda_a(y_i)$ is undefined, for $i = 1,\ldots,n-1$, and $\lambda_a(y_n) = c_{i_k}$. In case (a) it follows directly from the definition of λ_a that $y_1[a = c_{i_k}] \in \iota_\pi(\Omega_T)$. In case (b) we have amongst other chains the following inheritance chains in Ω_T:

$$
\begin{aligned}
&y_1 < y_2\\
&y_1 < y_2 < \cdots < y_n\\
&y_1 < y_2 < \cdots < y_n[a = c_{i_k}]\\
&\quad\vdots\\
&y_1 < y_2 < \cdots < y_m[a = c_{i_m}]
\end{aligned}
$$

$1 \leqslant n \leqslant m$. Hence, it follows from Definition 3.10 that $y_1[a = c_{i_k}] \in \iota_\pi(\Omega_T)$.

On the other hand, let $y_1[a = c_{i_k}] \in \iota_\pi(\Omega_T)$. We have again two cases. In case (a) we have that $y_1[a = c_{i_k}] \in \Theta|_K$, and in case (b) we have that $y_1[a = c_{i_k}] \notin \Theta|_K$ but $y_1[a = c_{i_k}] \in \iota_\pi(\Omega_T)$. In case (a) we have that $y_1^{a=c_{i_k}} \in S_{G_a}$; in case (b) we may construct paths corresponding to the inheritance chains given above; Algorithm 4.2 then yields the element $y_1^{a=c_{i_k}}$. Hence, we may conclude that $\iota_\pi(\Omega_T) \equiv S_{G_a}$. □

4.3. Transformation of an attribute graph to an i-tree

A clear disadvantage of applying the procedures `ConstructChains` and `PiConsistency` in Algorithm 4.1 for checking the π-consistency of a taxonomy and producing the inheritable conclusion set, is in combination their exponential nature. As has already been explained, this is due to the process of generating all inheritance chains in the procedure `ConstructChains`. This result stands in sharp contrast to the polynomial time complexity of single inheritance with exceptions in tree-like taxonomies for a given attribute a. Hence, it seems desirable to have some method available that is capable of transforming the i-graph of a given graph-like taxonomy to the i-tree of an equivalent tree-like taxonomy. Such a method indeed exists. In the following example we illustrate the basic idea.

EXAMPLE. Consider the taxonomy $T_1 = (N,\Theta,\ll,<)$ which is depicted in Figure 6. The taxonomy contains the following two attribute-value specifications:

$$
\begin{aligned}
&y_2[a = c_1]\\
&y_3[a = c_2]
\end{aligned}
$$

where a is assumed to be an attribute, and c_1 and c_2 are two distinct constants. As can be seen, $y_2[a = c_1]$ is an exception.

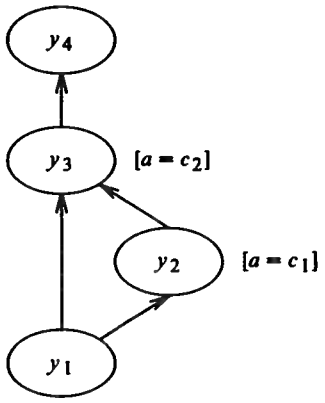

FIGURE 6. A graph-like taxonomy.

Now, consider the following subset of the entire set of inheritance chains Ω_{T_1}:

ω_1: $y_1 < y_2$
ω_2: $y_1 < y_3$
ω_3: $y_1 < y_2 < y_3$
ω_4: $y_1 < y_2[a = c_1]$
ω_5: $y_1 < y_3[a = c_2]$
ω_6: $y_1 < y_2 < y_3[a = c_2]$

From Definition 3.8 it follows that the inheritance chain ω_4 precludes both the chains ω_5 and ω_6, since y_2 is intermediary to the chains ω_2 and ω_3 according to Definition 3.7. The inheritable conclusion set $\iota_\pi(\Omega_{T_1})$ is equal to $\{y_1[a = c_1],\ y_2[a = c_1],\ y_3[a = c_2]\}$. Consider the taxonomy $T_2 = (N,\Theta,\ll,<')$, depicted in Figure 7, which differs from the taxonomy T_1 by the absence of the element $y_1 < y_3$ from the definition of the subclass relation. (Nevertheless, the element $y_1 < y_3$ is still satisfied by transitivity.) As a consequence, its set of inheritance chains does not contain the chain ω_6, but it does contain the inheritance chain ω_5. However, as in T_1, the latter chain is precluded by ω_4, and the inheritable conclusion sets of the two taxonomies are equal. It should be noted that the former taxonomy T_1 is graph-like, while the graph representation of the latter taxonomy T_2 is a tree. □

We shall now pay attention to an algorithm that transforms a general *i*-graph of a graph-like taxonomy into an *i*-tree of an equivalent tree-like taxonomy. The first part of the algorithm sorts the vertices in the directed attribute graph G_a into a topological order, starting with the most general class τ, where arcs are assumed to be traversed in a direction reverse to the one indicated. The reverse direction expresses the superclass, instead of the subclass interpretation of arcs. Such a topological sort is possible, since the directed graph is acyclic. However, the superclass relation in a frame taxonomy defines a partial order on the classes, not a total order as is yielded by the topological sort. This is corrected by the second part of the algorithm, which also handles attribute-value specifications in a correct way. This part of the algorithm constructs a breadth-first spanning tree S of the attribute graph G by traversing the graph in the reverse direction. The resulting tree is therefore called the *reverse breadth-first spanning tree*. The branches in the resulting *i*-tree still have a direction that is the

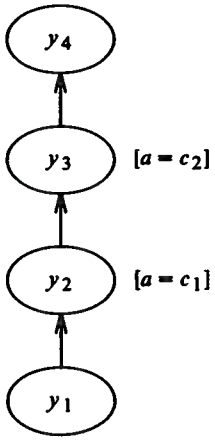

FIGURE 7. Tree-like taxonomy equivalent to the taxonomy in Figure 6.

same as in the original graph, since the reverse breadth-first traversal is merely performed for selecting branches to obtain a tree which has a corresponding tree-like taxonomy. Consequently, depth-first search in a reverse direction would serve as well. If we take the vertex corresponding to the most general class $\top$, the source, as a starting point for breadth-first search, then we have that in the resulting directed tree there exists a path from each vertex to the root of the tree. However, since the way this tree has been constructed is based on the traversal order imposed by breadth-first search, and not by examining attribute-value specifications in the original graph G, some of the branches in the tree may have been selected wrongly. This means that the set of attribute-value specifications obtained from applying Algorithm 4.2 on the resulting i-tree does not correspond to the inheritable conclusion set of the graph-like taxonomy. We therefore proceed by reconsidering these choices of arcs by adding successively each of the arcs in G not in S as edges to the underlying graph H of S. Initially, the number of these arcs is equal to the cycle rank of the underlying graph of G. The cycle created by each addition of an edge, occurring in the underlying graph of G but not in H, to H is examined, and based on the consideration of intermediary classes it is decided whether or not the arc in S corresponding to the edge added to H will be removed and replaced by another arc. In such a way, the original spanning tree S and its underlying graph H may be transformed. This procedure is repeated until all arcs have been examined. In case a decision has been reached for every cycle thus created, Algorithm 4.2 for single inheritance may be applied on the resulting spanning tree. However, if a particular cycle has been constructed where the decision criterion fails, we have a π-inconsistent taxonomy T: no suitable spanning tree exists in this case.

The order in which the arcs are being processed is important. Since inheritance chains essentially express the order in which inheritance of attribute-value specifications should take place, we have to impose an order on the arcs respecting the relative position of the vertices in the directed attribute graph. A suitable basis for such an ordering is obtained by the topological sort of the vertices in the graph mentioned above. The arcs are assumed to be ordered according to the ordering number of their first component. After this global sketch of the principal idea, let us now go into some more detail.

To examine the graph on the presence of conflicting attribute-value specifications, we first change the directed attribute graph of the taxonomy T into an undirected attribute graph, called the underlying attribute graph.

DEFINITION 4.3. Let $G_a = (V(G_a),R(G_a),\lambda_a)$ be an attribute graph of a taxonomy $T = (N,\Theta,\ll,<)$, for some $a \in A$, where A is the set of attributes in T. The *underlying attribute graph* H_a of G_a is the undirected graph $H_a = (V(H_a),E(H_a),\lambda_a)$, where $V(H_a) = V(G_a)$, and $E(H_a)$ is obtained by considering all arcs in $R(G_a)$ as edges in $E(H_a)$. Moreover, the function λ_a in H_a is taken from G_a. □

In particular, cycles in the underlying attribute graph H_a with vertices having attribute-value specifications are of interest to us, for these indicate potential causes of π-inconsistency. Therefore, in the following definition the notion of an attribute cycle is introduced. Such a cycle contains at least one attribute-value specification for a particular vertex in G_a, which, after adding direction to the edges in the cycle graph, has an out-degree equal to one.

DEFINITION 4.4. Let $G_a = (V(G_a),R(G_a),\lambda_a)$ be an attribute graph of a taxonomy $T = (N,\Theta,\ll,<)$, for some $a \in A$, and let $H_a = (V(H_a),E(H_a),\lambda_a)$ be the underlying attribute graph of G_a. A cycle $\Gamma = y_1,y_2,\ldots,y_n,z_q,z_{q-1},\ldots,z_1,y_1$, $n \geqslant 2$, $q \geqslant 0$, in H_a is called an *attribute cycle*, if $y_1,y_2,\ldots,y_n$ and $y_1,z_1,z_2,\ldots,z_q,y_n$ are distinct paths from y_1 to y_n in G_a, and if there exists at least one k, $1 < k < n$, such that $\lambda_a(y_k)$ is defined. Otherwise, the cycle Γ in H_a is called a *non-attribute cycle*. □

DEFINITION 4.5. Let G_a and H_a be defined as in the foregoing definition. An attribute cycle $\Gamma = y_1,y_2,\ldots,y_n,z_q,z_{q-1},\ldots,z_1,y_1$, $n \geqslant 2$, $q \geqslant 0$, in H_a is called *straight* if there exists at least one k, $1 < k < n$, such that $\lambda_a(y_k)$ is defined, and if for each j, $1 \leqslant j \leqslant q$, $\lambda_a(z_j)$ is undefined. □

DEFINITION 4.6. Let G_a and H_a be defined as in Definition 4.4. An attribute cycle $\Gamma = y_1,y_2,\ldots,y_n,z_q,z_{q-1},\ldots,z_1,y_1$, $n \geqslant 2$, $q \geqslant 0$, in H_a is called *ambiguous* if there exists at least one k, $1 < k < n$, such that $\lambda_a(y_k) = c$, and if there exists at least one j, $1 \leqslant j \leqslant q$, such that $\lambda_a(z_j) = d$. An ambiguous attribute cycle is called *arbitrary* if it is an ambiguous attribute cycle for which $\lambda_a(y_1)$ exists, or an ambiguous attribute cycle for which we have that for each k, $1 < k < n$, and for each j, $1 \leqslant j \leqslant q$, if both $\lambda_a(y_k)$ and $\lambda_a(z_j)$ are defined, we have $\lambda_a(y_k) = \lambda_a(z_j)$. □

Let us now consider the topological order on the vertices of a directed attribute graph. As we said above, the transformation of the spanning tree will be guided by this ordering.

DEFINITION 4.7. Let $G_a = (V(G_a),R(G_a),\lambda_a)$ be the directed attribute graph associated with the taxonomy $T = (N,\Theta,\ll,<)$, for some attribute $a \in A$. A *topological order* on the vertices $V(G_a) = \{y_1,y_2,\ldots,y_n\}$, $n \geqslant 1$, is a function τ: $V(G_a) \rightarrow \{1,2,\ldots,n\}$, such that if $(y_i,y_j) \in R(G_a)$, we have $\tau(y_j) < \tau(y_i)$. The *ordered attribute graph* $G_a^* = (V(G_a^*),R(G_a^*),\lambda_a^*)$ is obtained from G_a by attaching to each vertex $y_i \in V(G_a)$ an integer superscript equal to the ordering number $\tau(y_i)$. □

We finally are in a position to present the definition of the reverse breadth-first spanning tree.

DEFINITION 4.8. Let $G_a = (V(G_a),R(G_a),\lambda_a)$ be the directed attribute graph associated with the taxonomy $T = (N,\Theta,\ll,<)$, for some $a \in A$, and let $G_a^* = (V(G_a^*),R(G_a^*),\lambda_a^*)$ be the ordered attribute graph of G_a. Then, $S_a = (V(S_a),R(S_a),\lambda_a)$, where $V(S_a) = V(G_a^*)$, $R(S_a) \subseteq R(G_a^*)$, and λ_a is taken from G_a^* is a directed tree, called the *reverse breadth-first spanning tree* of G_a. □

By the application of the Definitions 4.1 to 4.8 it is possible to construct a spanning tree P_a for the attribute graph G_a, for some $a \in A$, which is an i-tree of a tree-like taxonomy, equivalent to the original graph-like taxonomy. This construction is described by the following algorithm.

ALGORITHM 4.3. Let $T = (N,\Theta,\ll,<)$ be a graph-like taxonomy, and let A be the set of attributes in T. Suppose that $G_a = (V(G_a),R(G_a),\lambda_a)$ is the associated i-graph of T, for some $a \in A$, and let G_a^* be the ordered attribute graph of G_a. In the algorithm the directed attribute graph G_a is transformed into G_a^* by a call to the function `TopologicalSort`. Furthermore, let $S_a = (V(S_a),R(S_a),\lambda_a)$ be the reverse breadth-first spanning tree of G_a, and let $H_a = (V(H_a),E(H_a),\lambda_a)$ be the underlying attribute graph of S_a. In the algorithm, the spanning tree S_a is obtained from the ordered attribute graph G_a^* by a call to the function `BFS`. Let $\Gamma = y_1,y_2,\ldots,y_n,z_q,z_{q-1},\ldots,z_1,y_1$, $n \geqslant 2$, $q \geqslant 0$, be a cycle obtained by adding some arc $(y_1,u) \in R(G_a^*) \setminus R(S_a)$ as an edge to H_a, where u is either equal to y_2 or to z_1. We suppose that $y_1,y_2,\ldots,y_n$, and $y_1,z_1,\ldots,z_q,y_n$ are distinct paths from y_1 to y_n in G_a. Let B be the set of arcs occurring in $R(G_a^*)$ but not in $R(S_a)$. Then, we assume that $s(B)$ is a sequence of elements (y_i^k,y_j^l), where for each $b_i = (y_p^k,y_q^l)$, $b_j = (y_r^m,y_s^n) \in s(B)$ we have that $i < j$ if $k < m$. A spanning tree P_a for G_a which is an i-tree of an equivalent tree-like taxonomy is obtained by the following algorithm.

```
proc Spanning-I-Tree(G_a)
  G_a^* ← TopologicalSort(G_a);
  S_a ← BFS(G_a^*);
  B ← R(G_a^*) \ R(S_a);
  consistent ← true;
  for each r ∈ s(B) while consistent do
    construct cycle Γ by adding the edge corresponding to r to H_a;
    if Γ is a straight attribute cycle then
      Transform(S_a,Γ)
    elsif Γ is an ambiguous cycle but not arbitrary then
      consistent ← false
    fi
  od;
  if consistent then
    report "constructed i-tree is P_a = (V(S_a),R(S_a),λ_a)"
  else report "taxonomy is π-inconsistent" fi
end
```

In the procedure `Spanning-I-Tree` an arc $r \in s(B)$ is processed in the procedure `Transform` if adding r as an edge to the underlying attribute graph H_a of S_a gives rise to a straight attribute cycle. In that case, the set $R(S_a)$ will be possibly modified in the procedure `Transform` as follows. In case the cycle Γ defined above is a straight attribute cycle, the attribute graph G_a contains at least one element $y_k^{a=c_k}$, $1 < k < n$. If $r \equiv (y_1,y_2)$, the procedure `Transform` will modify the spanning tree S_a by adding the arc r and removing (y_1,z_1); otherwise, it returns the spanning tree unmodified. If the cycle is an ambiguous cycle, an inconsistency has been encountered if it is not an arbitrary cycle. In case the cycle Γ is a non-attribute cycle, S_a does not need to be modified, and the arc r is just ignored. After leaving the for loop, the taxonomy turns out to be π-consistent if no non-arbitrary ambiguous cycle has been encountered. We then have obtained the spanning i-tree P_a; otherwise the taxonomy is π-inconsistent.

The set of attribute-value specifications, which constitutes the inheritable conclusion set of T can now be constructed by repeatedly applying Algorithm 4.2 on the resulting i-tree P_a. □

It should be noted that some arcs in the resulting i-tree P_a may only have been inserted into $R(P_a)$ since they have been traversed by breadth-first search, and, for an opposite reason, other arcs have not been included. These arcs that were not inserted into $R(P_a)$ all yield arbitrary ambiguous cycles, or non-attribute cycles in H_a. Thus, it makes no difference for the inheritable conclusion set if other arcs in the arbitrary cycles, or non-attribute cycles, had been chosen by reverse breadth-first search for insertion into $R(P_a)$ than the ones that actually have been admitted. We call i-trees which only

differ with respect to such arcs *similar*. Moreover, note that the actual order in which the arcs (y_i,y_j) in the sequence $s(B)$ having the same component y_i are processed in Algorithm 4.3 is unimportant.

Let us first give two examples of the application of the algorithm, before proceeding to the proof of its correctness. In the first example, we give a pictorial demonstration of the algorithm.

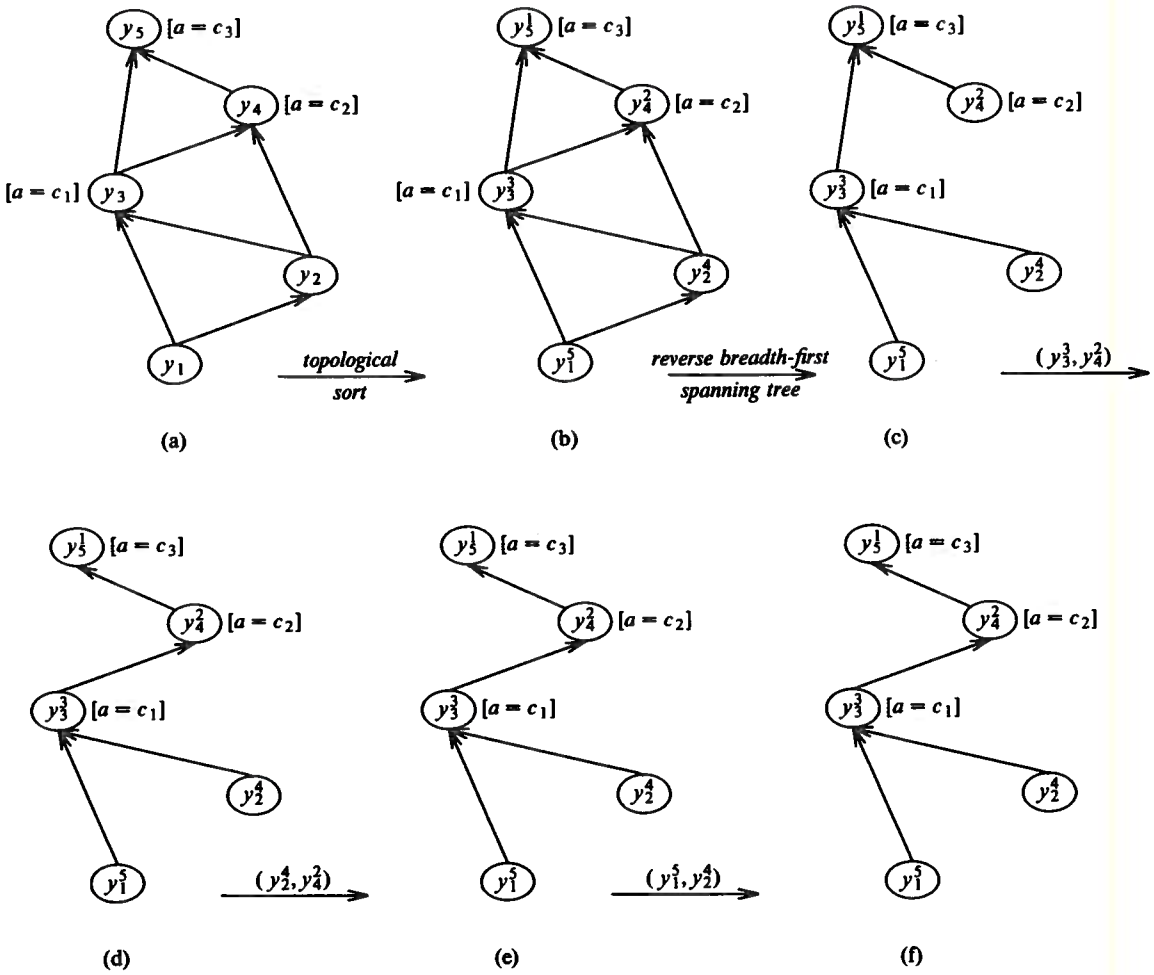

FIGURE 8. Example of application of Algorithm 4.3.

EXAMPLE. Consider the following taxonomy $T = (N,\Theta,\ll,<)$, where $I = \varnothing$, $K = \{y_1,y_2,y_3,y_4,y_5\}$, and $\Theta = \{y_3[a = c_1],\ y_4[a = c_2],\ y_5[a = c_3]\}$. Let the relation $<$ be defined as follows:

$$y_1 < y_2$$
$$y_1 < y_3$$
$$y_2 < y_3$$
$$y_2 < y_4$$

$y_3 < y_4$
$y_3 < y_5$
$y_4 < y_5$

The attribute graph representation $G_a = (V(G_a),R(G_a),\lambda_a)$ of T is shown in Figure 8.a. In Figures 8.b to 8.f the various steps through which this attribute graph is transformed into an i-tree are shown. It follows that the taxonomy is π-consistent. □

In the next example, we illustrate the use of the algorithm in a taxonomy containing an instance, along the same lines as discussed for the inheritable extension of a taxonomy in Section 3.2.

EXAMPLE. Consider the following taxonomy $T = (N,\Theta,\ll,<)$, where $I = \{x\}$, $K = \{y_1,y_2,y_3,y_4\}$ and $\Theta = \{y_2[a = c_1], y_4[a = c_2]\}$. Let the function $\ll$ and the relation $<$ be defined as follows:

$x \ll y_1$
$y_1 < y_2$
$y_2 < y_3$
$y_3 < y_4$
$y_1 < y_3$

The inheritable conclusion set $\iota_\pi(\Omega_T)$ is equal to $\{y_1[a = c_1], y_2[a = c_1], y_3[a = c_2], y_4[a = c_2]\}$.

Let us now reconsider the i-graph $G_a = (V(G_a),R(G_a),\lambda_a)$, of the taxonomy T, and construct the i-tree P_a for the attribute a. The i-graph of T is as follows: $V(G_a) = \{y_1,y_2,y_3,y_4\}$, $R(G_a) = \{(y_1,y_2), (y_1,y_3), (y_2,y_3), (y_3,y_4)\}$, and $\lambda_a(y_2) = c_1$, $\lambda_a(y_4) = c_2$. The underlying attribute graph of G_a only contains a single cycle, which is y_1,y_2,y_3,y_1. The ordered attribute graph is equal to $G_a^* = (V(G_a^*),R(G_a^*),\lambda_a^*)$, where $V(G_a^*) = \{y_1^4,y_2^3,y_3^2,y_4^1\}$, $R(G_a^*) = \{(y_1^4,y_2^3), (y_1^4,y_3^2), (y_2^3,y_3^2), (y_3^2,y_4^1)\}$, and $\lambda_a^*(y_2^3) = c_1$, $\lambda_a^*(y_4^1) = c_2$. Suppose that the reverse breadth-first spanning tree S_a contains all vertices and arcs in G_a^*, except the arc (y_1^4,y_2^3). Algorithm 4.3 then transforms the tree S_a into an i-tree $P_a = (V(G_a^*),R(P_a),\lambda_a)$, where $R(P_a) = R(G_a^*) \setminus \{(y_1^4,y_3^2)\}$. If we carry out Algorithm 4.2 for single inheritance on the i-tree P_a we obtain the set $\{y_1^{a=c_1}, y_2^{a=c_1}, y_3^{a=c_2}, y_4^{a=c_2}\}$, which corresponds to $\iota_\pi(\Omega_T)$. For the instance x we therefore have the element $x^{a=c_1}$. □

We next prove the correctness of Algorithm 4.3 described above.

LEMMA 4.3. *Let $T = (N,\Theta,\ll,<)$ be a graph-like taxonomy. Let A be the set of attributes in T, and let Ω_T be the set of inheritance chains in T. Then, the set of attribute-value specifications, obtained by the application of Algorithm 4.3 on the set of associated i-graphs $\{G_a \mid a \in A\}$, for each $y \in V(G_a)$, corresponds to $\iota_\pi(\Omega_T)$.*

PROOF. Let $G_a = (V(G_a),R(G_a),\lambda_a)$ be the associated i-graph of the taxonomy T, defined for an arbitrary attribute $a \in A$, and let $G_a^* = (V(G_a^*),R(G_a^*),\lambda_a^*)$ be the ordered attribute graph of G_a. Suppose that $S_a = (V(S_a),R(S_a),\lambda_a)$ is the reverse breadth-first spanning tree obtained for the i-graph G_a as described in Algorithm 4.3, and let at each step in the algorithm H_a be the underlying attribute graph of S_a.

First, consider the case in which G_a^* is an i-tree. Then, Algorithm 4.3 terminates with a spanning tree P_a that is identical to G_a^*, since the sequence $s(B)$ is empty. Second, consider the case in which G_a^* is an i-graph, but no i-tree. The proof will be by induction on the length of the sequence $s(B)$. First, consider the case where $|s(B)| = 1$. Let again S_a be the initial breadth-first spanning tree of G_a^*. Adding the arc r from the sequence $s(B)$ as an edge to H_a yields a cycle $\Gamma = y_1,y_2,\ldots,y_n,z_q,z_{q-1},\ldots,z_1,y_1$, such that $y_1,y_2,\ldots,y_n$ and $y_1,z_1,z_2,\ldots,z_q,y_n$ are distinct paths in G_a^*. Let $Q_a = (V(Q_a),R(Q_a),\lambda_a')$ be the directed attribute graph associated with the

cycle Γ, such that $V(Q_a) = \{y_1, \ldots, y_n, z_1, \ldots, z_q\}$, and $R(Q_a) = \{(y_1,y_2), \ldots, (y_{n-1},y_n), (y_1,z_1), (z_1,z_2), \ldots, (z_{q-1},z_q), (z_q,y_n)\}$, where λ_a': $V(Q_a) \rightarrow C$ is the function λ_a^*, restricted to the domain $V(Q_a)$. Let T_{Q_a} be the graph-like taxonomy associated with the directed attribute graph Q_a. Furthermore, let $Y_a = (V(Y_a), R(Y_a), \lambda_a')$ be a directed attribute tree defined by $V(Y_a) = V(Q_a)$, $R(Y_a) = R(Q_a) \setminus \{(y_1,z_1)\}$ where λ_a' is taken from Q_a. It suffices to only consider the directed attribute graph Q_a associated with each created cycle Γ, since this is the only place in the algorithm were a decision is made with respect to the possible inheritance of multiple constants by the various vertices y_1. We have three cases. First, the cycle Γ can be a straight attribute cycle, secondly, Γ can be an ambiguous attribute cycle, and, finally, Γ can be a non-attribute cycle.

If the cycle Γ is a straight attribute cycle, the directed attribute graph Q_a is examined to determine whether the removal of the particular arc r from G_a, and from Q_a yielding the tree Y_a, is justified with regard to the interpretation function ι_π. Suppose that the straight attribute cycle contains some vertices for which we have elements $y_k^{a=c_{i_k}}$, $1 < k < n$, then it is justified to remove the arc (y_1,z_1), since no vertex z_j contributes attribute-value specifications directly to the vertex y_1. Application of Algorithm 4.2 on the vertices of the tree Y_a produces the following set of attribute-value specifications:

$$S_{Y_a} = \{y_1^{a=c_{i_1}}, \ldots, y_l^{a=c_{i_l}}, y_{l+1}^{a=c_{i_m}}, \ldots, y_m^{a=c_{i_m}}\}$$

where $1 \leqslant l \leqslant m \leqslant n$. From the associated taxonomy T_{Q_a} of the directed attribute graph Q_a, we can construct the following set of inheritance chains Ω:

$$\begin{array}{l}
y_1 < \cdots < y_l[a = c_{i_l}] \\
y_2 < \cdots < y_l[a = c_{i_l}] \\
\quad \vdots \\
y_l[a = c_{i_l}] \\
y_1 < \cdots < y_l < \cdots < y_m[a = c_{i_m}] \\
y_2 < \cdots < y_l < \cdots < y_m[a = c_{i_m}] \\
\quad \vdots \\
y_m[a = c_{i_m}]
\end{array}$$

where we have only listed inheritance chains having an attribute-value specification at the end. The inheritable conclusion set $\iota_\pi(\Omega)$ is equal to $\iota_\pi(\Omega) = \{y_1[a = c_{i_1}], \ldots, y_l[a = c_{i_l}], y_{l+1}[a = c_{i_m}], \ldots, y_m[a = c_{i_m}]\}$. Hence, $S_Q \equiv \iota_\pi(\Omega)$.

In case we have an ambiguous attribute cycle Γ, the taxonomy T may be π-inconsistent. This follows from examining the inheritance chains which can be constructed for the taxonomy T_{Q_a}.

$$\begin{array}{l}
y_1 < \cdots < y_l[a = c_{i_l}] \\
y_2 < \cdots < y_l[a = c_{i_l}] \\
\quad \vdots \\
y_l[a = c_{i_l}] \\
y_1 < \cdots < y_l < \cdots < y_m[a = c_{i_m}] \\
y_2 < \cdots < y_l < \cdots < y_m[a = c_{i_m}] \\
\quad \vdots \\
y_m[a = c_{i_m}]
\end{array}$$

$$
\begin{array}{l}
y_1 < z_1 < \cdots < z_p[a = d_{i_p}] \\
z_1 < \cdots < z_p[a = d_{i_p}] \\
z_2 < \cdots < z_p[a = d_{i_p}] \\
\quad \cdot \\
\quad \cdot \\
z_p[a = d_{i_p}] \\
z_1 < \cdots < z_q[a = d_{i_q}] \\
z_2 < \cdots < z_q[a = d_{i_q}] \\
\quad \cdot \\
\quad \cdot \\
z_q[a = d_{i_q}]
\end{array}
$$

Since the inheritance chains $y_1 < \cdots < z_p[a = c_{i_p}]$ and $y_1 < \cdots < y_l[a = c_{i_l}]$ do not preclude each other, we might have a π-inconsistent taxonomy. In case the ambiguous cycle turns out to be arbitrary, the taxonomy will be π-consistent, since only a single attribute-value specification $y_1[a = c]$ is obtained from the inheritance chains. Otherwise, the algorithm terminates by reporting that the taxonomy is π-inconsistent.

In the last case, when Γ is a non-attribute cycle, the choice made by the breadth-first strategy for a particular path from the root to each vertex in the cycle does not affect the result obtained by Algorithm 4.2. The remainder of the proof follows from Lemma 4.2.

Now, let us assume that the algorithm is correct for the case that $|s(B)| = n$. We next prove that from this assumption it follows that the algorithm is correct for $|s(B)| = n + 1$. Let us add the arc $b_{n+1} = (y_1,u)$ as an edge to H_a, we then have again obtained a cycle Γ, similar to the one given above. From the order imposed on the arcs, it follows that we only have to consider inheritance chains having the class corresponding to the vertex y_1 at their beginning. The argumentation then goes in a similar way as for the case that $|s(B)| = 1$. □

THEOREM 4.1. *Let $T = (N,\Theta,\ll,<)$ be a graph-like taxonomy, and let A be the set of attributes in T. Application of Algorithm 4.3 on the associated i-graph, for each $a \in A$, terminates in polynomial time either with a set of attribute-value specifications which corresponds to $\iota_\pi(\Omega_T)$, or by reporting π-inconsistency of the taxonomy T.*

PROOF. Termination of Algorithm 4.3 is shown by noting that $|s(B)|$ equals the cycle rank κ of the underlying attribute graph of G_a. At each step in the execution of the for loop in the procedure `Spanning-I-Tree` exactly one arc r from this sequence is processed and possibly transformed in the procedure `Transform`; if all arcs have been processed or an inconsistency has been encountered at a particular point, the for loop terminates.

Let $|V(G_a)| = n$. The initial steps in the algorithm consist of performing a topological sort of the vertices in the graph and reverse breath-first search, both of which can be done in at most $O(n^2)$ steps. For any directed attribute graph we have from the properties of the cycle rank of a directed acyclic graph that $\kappa \leqslant \frac{1}{2}n^2 - \frac{3}{2}n + 1 = O(n^2)$. Checking whether the cycle constructed in the procedure `Transform` is either a straight cycle, an ambiguous cycle, or a non-attribute cycle takes at most $O(n)$ time. We conclude that the overall worst-case time complexity of the algorithm is $O(n^3)$. Finally, the correctness of the procedure to transform an *i*-graph into an *i*-tree follows from Lemma 4.3. □

5. Conclusions

In this article we have analysed inheritance of attribute values in various kinds of taxonomies, and more in particular multiple inheritance with exceptions. In the first part of this paper, we analysed multiple inheritance in a mathematical way using inheritance chains. It was shown where inconsistencies arise in these taxonomies, and several techniques for checking the consistency of a taxonomy have been discussed. In the second part of the paper, we studied inheritance from an algorithmic point of view, and developed an algorithm for consistency checking in a graph-like taxonomy. A practical result was obtained concerning the transformation of a graph-like taxonomy to an equivalent tree-like taxonomy, making the former amenable to single inheritance with exceptions. It was shown that such an equivalent tree-like taxonomy could only be constructed in a graph-like taxonomy which is π-consistent.

We have focussed on inheritance of attribute values by classes. A possible extension to the frame formalism treated in this paper is the inheritance of structured data by classes and instances. This situation occurs when attribute-type specifications of the form $x(a{:}\tau)$ are defined, where τ is the name of a class. The artificial distinction between instances and constants can then be abandoned. Although we have not explicitly discussed these cases, the techniques developed in this paper form a suitable framework for analysing this extended frame formalism as well.

Acknowledgements

I like to thank Linda van der Gaag for her valuable comments on drafts of this paper. She actually suggested me to consider the relationship between spanning trees and multiple inheritance when we were enjoying a Gorgonzola pizza at our local pizzeria.

6. References

[Bobrow83] D.G. Bobrow, M.J. Stefik (1983). *The LOOPS Manual,* Xerox Corporation, Palo Alto, California.

[Bobrow88] D.G. Bobrow, L.G. DeMichiel, R.P. Gabriel, et al. (1988). *Common Lisp Object System Specification,* Xerox Corporation, Palo Alto, California.

[Cardelli84] L. Cardelli (1984). A Semantics of multiple inheritance, in: Kahn, MacQueen, and Plotkin (eds.), *Semantics of data types,* Lecture Notes in Computer Science 173, Springer Verlag, Berlin.

[KC88] Knowledge Craft, CRL Technical Manual (1988). Carnegie Group Inc, Pittsburgh.

[Fikes85] R. Fikes, T. Kehler (1985). The role of frame-based representation in reasoning, *Communications of the ACM, vol. 28, number 9, p. 904-920.*

[Lewis81] H.R. Lewis, C.H. Papadimitriou (1981). *Elements of the theory of computation,* Prentice-Hall, Englewood Cliffs.

[Minsky75] M. Minsky (1975). A framework for representing knowledge, in: P.H. Winston (ed.), *The Psychology of Computer Vision,* McGraw-Hill, New York.

[NO88] Nexpert Object Fundamentals, Neuron Data Inc, Palo Alto, California.

[Smolka87] G. Smolka, H. Ait-Kaci (1987). *Inheritance hierarchies: semantics and unification,* MCC, Technical Report AI-057-87.

[Touretzky86] D.S. Touretzky (1986). *The Mathematics of Inheritance Systems,* Pitman, London.

[Wilson85] R.J. Wilson (1985). *Introduction to graph theory,* Longman, New York.